# Demand Restoring Algorithm: Step Description

This step defines how sales history should be treated as a demand (see Ошибка: источник перекрёстной ссылки не найденОшибка: источник перекрёстной ссылки не найден). Particularly, the main purpose of this transformation is to provide downstream steps with as most as possible correct information regarding unconstrained demand observed in the past. Thus, there will be several outputs from this step:

1. Mark SKU – store – day combinations, in which sales volumes do not reveal true demand information;
2. Restore demand volumes for those triples SKU/Store/Day from the point 1.

## Code Realization Requirements

Use python

# Input Data

The initial data for the demand recovery process is listed below. These tables should be present in the system before this step for example in STG or in DDS area (see data requirements document).

## IN\_PRODUCT\_ATTR

(see data requirements)

Only information about seasonal indicator is needed if it’s available.

## IN\_SALES

Sales information regarding the past till last known day of the history.

|  |  |
| --- | --- |
| DDS.IN\_SALES | |
| Column Name | Description |
| **PRODUCT\_ID** | Product ID (the lowest level of the product hierarchy) |
| **LOCATION\_ID** | Location ID |
| **PERIOD\_DT** | Date of sales (calendar day) |
| **SALES\_QTY** | Total sales in units per day (w/o returns) |
| **PROMO\_ID** | Promo ID |
| **PROMO\_FLG** | Promo flag |

## IN\_SELL\_IN

Sell-in information regarding the past till last known day of the history.

|  |  |
| --- | --- |
| DDS.IN\_SELL\_IN | |
| Column Name | Description |
| **PRODUCT\_ID** | Product ID (the lowest level of the product hierarchy) |
| **LOCATION\_ID** | Location ID |
| **CUSTOMER\_ID** | Customer ID |
| **DISTR\_CHANNEL\_ID** | Distribution CHANNEL ID |
| **LOCATION\_TO\_ID** | Location to Id |
| **PERIOD\_DT** | Date of SELL-IN (calendar day) |
| **ORDER\_QTY** | Total ORDER in units per day (w/o returns) |
| **ORDER\_AMOUNT** | Total ORDER revenue (with VAT) per day (w/o returns) |
| **SHIPMENTS\_QTY** | Total SHIPMENT in units per day (w/o returns) |
| **SHIPMENTS\_AMOUNT** | Total SHIPMENT revenue (with VAT) per day (w/o returns) |
| **INVOICE\_QTY** | Total INVOICE in units per day (w/o returns) |
| **INVOICE\_AMOUNT** | Total INVOICE revenue (with VAT) per day (w/o returns) |
| **RETURNS\_QTY** | Total returns in units per day |
| **RETURNS\_AMOUNT** | Total returns amount (with VAT) per day |
| **PROMO\_FLG** | Promo flag |
| **PROMO\_ID** | Promo ID |
| **COST** | Unit cost |

## IN\_SELL\_OUT

Sell-out information regarding the past till last known day of the history.

|  |  |
| --- | --- |
| DDS.IN\_SELL\_OUT | |
| Column Name | Description |
| **PRODUCT\_ID** | Product ID (the lowest level of the product hierarchy) |
| **LOCATION\_ID** | Location ID |
| **LOCATION\_FROM\_ID** | Location ID |
| **CUSTOMER\_ID** | Customer ID |
| **DISTR\_CHANNEL\_ID** | Distribution CHANNEL ID |
| **LOCATION\_TO\_ID** | Location to Id |
| **PERIOD\_DT** | Date of SELL-OUT (calendar day) |
| **ORDER\_QTY** | Total ORDER in units per day (w/o returns) |
| **ORDER\_AMOUNT** | Total ORDER revenue (with VAT) per day (w/o returns) |
| **SHIPMENTS\_QTY** | Total SHIPMENT in units per day (w/o returns) |
| **SHIPMENTS\_AMOUNT** | Total SHIPMENT revenue (with VAT) per day (w/o returns) |
| **INVOICE\_QTY** | Total INVOICE in units per day (w/o returns) |
| **INVOICE\_AMOUNT** | Total INVOICE revenue (with VAT) per day (w/o returns) |
| **RETURNS\_QTY** | Total returns in units per day |
| **RETURNS\_AMOUNT** | Total returns amount (with VAT) per day |
| **PROMO\_FLG** | Promo flag |
| **PROMO\_ID** | Promo ID |
| **COST** | Unit cost |

## IN\_STOCK

Inventory history data containing the following fields is used as an input:

|  |  |
| --- | --- |
| DDS.IN\_STOCK | |
| Column Name | Description |
| **PRODUCT\_ID** | Product ID (the lowest level of the product hierarchy) |
| **LOCATION\_ID** | Location ID |
| **PERIOD\_DT** | Date of sales (calendar day) |
| **STOCK\_QTY** | Stock qty |

If there is no inventory information in the PRODUCT\_ID | LOCATION\_ID | PERIOD\_DT combination, the stock value is set to zero.

## IN\_PROMO

Promo history data containing the following fields is used as an input.

|  |  |
| --- | --- |
| DDS.IN\_PROMO | |
| Column Name | Description |
| **PROMO\_ID** | Id promo |
| **PRODUCT\_ID** | Product ID (the lowest level of the product hierarchy) |
| **LOCATION\_ID** | Location ID |
| **CUSTOMER\_ID** | Customer ID |
| **DISTR\_CHANNEL\_ID** | Distribution Channel ID |
| **PERIOD\_START\_DT** | Promo start date |
| **PERIOD\_END\_DT** | Promo end date |
| **PROMO\_TYPE** | Promo type ID |

## FORECAST\_FLAG

Product/location lifecycle history containing the following fields is used as an input:

|  |  |
| --- | --- |
| DM.FORECAST\_FLAG | |
| Column Name | Description |
| **PRODUCT\_ID** | Product ID (the lowest level of the product hierarchy) |
| **LOCATION\_ID** | Location ID |
| CUSTOMER\_ID | Customer ID |
| DISTR\_CHANNEL\_ID | Distribution Channel ID |
| **PERIOD\_START\_DT** | Period start date |
| **PERIOD\_END\_DT** | Period end date |
| **STATUS** | One of the following status: maturity, new, end-of-life (only active-like periods) |

FORECAST\_FLAG must be calculated before DEMAND\_RESTORATION.

## RESTORED DEMAND

Values are stored at previous runs.

|  |  |
| --- | --- |
| DEMAND\_RESTORED \_<postfix for considered target variable type> | |
| Field | Description |
| **PRODUCT\_ID** | Product ID |
| **LOCATION\_ID** | Location ID |
| **CUSTOMER\_ID** | Customer ID |
| **DISTR\_CHANNEL\_ID** | Distribution Channel ID |
| **PERIOD\_DT** | Date of sales (calendar day) |
| **TGT\_QTY** | Actual quantity values of target variable |
| **TGT\_QTY\_R** | Restored demand for chosen target variable |
| **STOCK\_QTY** | Stock qty (BOP) |
| **PROMO\_FLG** | 1|promo event was active  0|no promo event |
| **PROMO\_TYPE** | Promo\_type id or missing if there is no promo event |
| **PROMO\_ID** | Promo ID |
| **DEFICIT\_FLG1** | 1|primary deficit occurred  0|no primary deficit |
| **DEFICIT\_FLG2** | 1|secondary deficit occurred  0|no secondary deficit |
| **COUNT\_NONDEFICIT\_DAYS** | The number of non-deficit days (primary deficit) that were included in the window of **DR\_PERIOD\_LENGTH** previous calendar days |
| **MEAN** | Average number of sales values for the previous **DR\_OBS\_NUM** observations without primary deficit () within particular group of days |
| **STD** | Standard deviation of sales values for the previous **DR\_OBS\_NUM** observations without primary deficit ( within particular group of days |

## CONFIG\_PARAMETERS

## STATIC GLOBAL PARAMTERS

These parameters are changed by user in ad-hoc manner, i.e. usually they are the same for different runs.

|  |  |  |
| --- | --- | --- |
|  | | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
| INITIAL\_GLOBAL parameters init | | |
| Column Name | | Description |
| **IB\_MAX\_DT** | | Maximal date which is used for data preparation (e.g. 01/01/2100) |
| **IB\_UPDATE\_HISTORY\_DEPTH** | | Number of days of historical information that should be considered within this step running, i.e. only dates since(>=) **IB\_HIST\_END\_DT - IB\_UPDATE\_HISTORY\_DEPTH** should be used within step |
| **IB\_HIST\_START\_DT** | | Minimal date that should be present in Demand Restored. |

## Dynamic GLOBAL parameters

These parameters are changed from run to run depending on current date, present data etc.

|  |  |
| --- | --- |
| INITIAL\_GLOBAL parameters init | |
| Column Name | Description |
| **IB\_HIST\_END\_DT** | Last known date (i.e. sales and stock information is known) |

## CONFIG FILES

The following config csv-files are used within demand restoration step:

## DR\_PARAMETERS.csv

The following config parameters are used within demand restoration step. All these parameters must be present in a separate csv-config file (DR\_PARAMETERS.csv)

|  |  |
| --- | --- |
| CONFIG.DR\_PARAMETERS | |
| Column Name | Description |
| **DR\_OBS\_NUM** | Maximal amount of observations in the past to be used for calculation a demand value. |
| **DR\_LIFECYCLE\_MARGIN** | Length (in days) of the period outside product/location lifecycle for which demand must be restored |
| **DR\_PERIOD\_LENGTH** | Length of the period (in calendar days), sales in which should be used for demand restoration for particular day |
| **DEF\_INV\_TRSHD** | Minimal stock volume in a day when deficit can be observed |
| **DEF\_QTY\_TRSHD** | Minimal sales volume in a day when deficit can be observed |
| **MIN\_SALES\_QTY\_DAY** | Minimal total value of sales that can be observed in a location per day |
| **MIN\_PROLONG\_HIST\_MONTH** | Minimal number of months in sales history for prolong history algorithm activation, default = 3 |
| **MAX\_PROLONG\_HIST\_MONTH** | Maximal number of months in sales history for prolong history algorithm activation, default = 24 |
| **HIGH\_TURNOVER\_TRSHD** | If mean demand is more that this threshold than quadruple is considered as one with high turnover. |
| **MIN\_ND\_DAYS** | Minimal number of non-deficit days for quadruple for restoring demand |

## Seasonal\_flag\_config.csv.

This file contains a field name and a list of values in the client data which are responsible for seasonal products differentiating.

|  |  |
| --- | --- |
| **FromSeasonalFlag** | **ToSeasonalFlag** |
| SEASONAL\_INDICATOR\_CD | * seasonal\_flag |
| EA | 1 |
| NY | 1 |
| BA | * 1 |
| XX | 1 |
| YY | 1 |

## TGT\_VAR\_CONFIG.csv

Target variables config file:

|  |  |  |
| --- | --- | --- |
| **CONFIG.TGT\_VAR\_CONFIG** | | |
| **Column Name** | **Description** | **Example** |
| **tgt\_type** | One of 3 types of the target variable:   * SELLIN – means CPG sales to its customer, * SELLOUT – means CPG’s customer sales to their clients, * POS – means sales in the point of sales, can be relevant for both Retailer and CPG | POS |
| **value\_src** | Name of the target variable from the source table. It should be quantity of sales.  Feasible values: INVOICE\_QTY, SALES\_QTY, SHIPMENT\_QTY, ORDER\_QTY. | SALES\_QTY |
| **act\_flag** | Activity flag, whether this target variable is needed to be forecasted. Feasible values: 0 or 1 | 1 |
| **dr\_scen** | Demand restoration scenario for target variable:   * Scenario 0 (parameter = 0). Demand restoration isn’t needed. * Scenario 1 (parameter = 1). Retail scenario to a greater extent. Demand restoration is performed based on stock data. Demand extending/prolongation for short seasonal products isn’t performed. * Scenario 2 (parameter = 2). CPG scenario to a greater extent. Only demand extending/prolongation for short seasonal products is performed. * Scenario 3 (parameter = 3). Scenario 1 and Scenario 2 are performed sequentially. | 3 |
| **link\_with\_stock** | Flag, whether this target variable is linked with provided stock data. Feasible values: 0 or 1 | 1 |
| **link\_with\_promo** | Flag, whether this target variable is linked with provided promo data. Feasible values: 0 or 1 | 0 |
| **link\_with\_price** | Flag, whether this target variable is linked with provided price data. Feasible values: 0 or 1 | 1 |
| **ts\_product\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 8, which means PRODUCT\_ID | 7 |
| **ts\_location\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 6, which means LOCATION\_ID | 1 |
| **ts\_customer\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 6, which means CUSTOMER\_ID | 5 |
| **ts\_distr\_channel\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 3, which means DISTR\_CHANNEL\_ID | 1 |
| **ts\_time\_lvl** | Accumulation level for ML ABT by time hierarchy, default value is WEEK.2, which means weeks began from Monday | MONTH |
| **ml\_product\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 8, which means PRODUCT\_ID | 7 |
| **ml\_location\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 6, which means LOCATION\_ID | 5 |
| **ml\_customer\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 6, which means CUSTOMER\_ID | 4 |
| **ml\_distr\_channel\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 3, which means DISTR\_CHANNEL\_ID | 1 |
| **ml\_time\_lvl** | Accumulation level for ML ABT by time hierarchy, default value is WEEK.2, which means weeks began from Monday | WEEK.2 |
|  |  |  |

|  |  |
| --- | --- |
|  | |
|  |  |
|  |  |

## Other Dependencies

Demand restoration algorithm can be run as for the whole historical period as well as for last n days ago, historical period depth handling should be performed using UPDATE\_PERIOD\_DEPTH parameter.

# Algorithm Definition

Assumptions for the demand restoration algorithm:

* All steps below are performed only for one target variable, which is based on chosen target type from configuration-file “tgt\_var\_config.csv” by filter **act\_flag** = 1.
* If more than one target variables should be forecasted and they were activated in config file, than result of this step will be a few tables, each of them related to own target variable.
* Name of the output tables of demand restoration algorithm should include postfix with type of target variable based on config file: \_SELLIN, \_SELLOUT or \_POS.

## Prepare sales and demand data

**Inputs:** FORECAST\_FLAG, IN\_SALES **or** IN\_SELL\_IN **or** IN\_SELL\_OUT, DEMAND\_RESTORED =DEMAND\_RESTORED\_<tgt\_type>.

**Transformation algorithm:**

1. Read **source\_kpi\_table** from tgt\_var\_config.csv (it is defined for particular tgt\_type).
2. Check whether previous version of DEMAND\_RESTORED table is available (is stored at previous step)

If not then parameter **IB\_UPDATE\_HISTORY\_DEPTH[[1]](#footnote-1) = 0 for all steps below.**

1. Select rows with status “active” from FORECAST\_FLAG table.
2. Extend period of lifecycle for each pair: add DR\_LIFECYCLE\_MARGIN to PERIOD\_END\_DT column, and get minimum PERIOD\_END\_DT = min (PERIOD\_END\_DT+ DR\_LIFECYCLE\_MARGIN, HISTORY\_END).
3. Select only those dates that are needed to update DR (since (>=) **IB\_HIST\_END\_DT - IB\_UPDATE\_HISTORY\_DEPTH** moment**)**

SELECT \*FROM 2 WHERE (**IB\_UPDATE\_HISTORY\_DEPTH**<=0

**OR**

period\_start\_dt **>= IB\_HIST\_END\_DT - IB\_UPDATE\_HISTORY\_DEPTH - DR\_PERIOD\_LENGTH OR**

period\_end\_dt **>= IB\_HIST\_END\_DT - IB\_UPDATE\_HISTORY\_DEPTH - DR\_PERIOD\_LENGTH)**

**and then**

Transpose all periods between

If **IB\_UPDATE\_HISTORY\_DEPTH<=0 then**

MAX(PERIOD\_START\_DT;

**IB\_HIST\_START\_DT- DR\_PERIOD\_LENGTH)**

If **IB\_UPDATE\_HISTORY\_DEPTH>0 then**

MAX(PERIOD\_START\_DT;

**(IB\_HIST\_END\_DT – MAX(IB\_UPDATE\_HISTORY\_DEPTH; 0) - DR\_PERIOD\_LENGTH);**

**IB\_HIST\_START\_DT- DR\_PERIOD\_LENGTH)**

and MIN(**PERIOD\_END\_DT, IB\_HIST\_END\_DT**)

to column PERIOD\_DT, select only distinct quintuple product | location | customer | distribution channel | day.

1. The table from previous step left join **source\_kpi\_table** based on **source\_kpi\_table** key columns: product | location | customer | distribution channel | day or product | location | day.
   1. Read **value\_src** from config-file for particular tgt\_type.
   2. Add **value\_src** column to output table, name the column as TGT\_QTY.
   3. Fill missing for TGT\_QTY as 0

For example:

SELECT 4.\*,

COALESCE(**source\_kpi\_table.value\_src, 0) as TGT\_QTY,**

**source\_kpi\_table.PROMO\_ID,**

**source\_kpi\_table.PROMO\_FLG,**

**‘ ‘ as PROMO\_TYPE**

FROM 4.

LEFT JOIN **sourc\_kpi\_table on <key\_columns>**

Output: As a result of this step, a table of the following structure is constructed, *T1*

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| **PRODUCT\_ID** | Product ID (the lowest level of the product hierarchy) |
| **LOCATION\_ID** | Location ID |
| **CUSTOMER\_ID** | Customer ID |
| **DISTR\_CHANNEL\_ID** | Distribution Channel ID |
| **PERIOD\_DT** | Date of sales (calendar day) |
| **TGT\_QTY** | Actual quantity values of target variable |
| **PROMO\_ID** | Promo id from sales/sellin/sellout table |
| **PROMO\_FLG** | Promo\_flg from sales/sellin/sellout table |
| **PROMO\_TYPE** | Missing columns for next steps |

## Adding stock data

**Inputs:** IN\_STOCK, T1

**Transformation algorithm:**

1. The table from previous step left join IN\_STOCK[[2]](#footnote-2) table on product | location | day and fill missing for STOCK\_QTY as 0, whether for considered **tgt\_type** the value of the field **link\_with\_stock** equals 1 in the config file “tgt\_var\_config.csv”. Else STOCK\_QTY should equals missing.

ASSUMPTION: there is only one CUSTOMER\_ID/DISTR\_CHANNEL\_ID for each LOCATION\_ID

Output: As a result of this step, a table of the following structure is constructed, *T2*

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| **PRODUCT\_ID** | Product ID (the lowest level of the product hierarchy) |
| **LOCATION\_ID** | Location ID |
| **CUSTOMER\_ID** | Customer ID |
| **DISTR\_CHANNEL\_ID** | Distribution Channel ID |
| **PERIOD\_DT** | Date of sales (calendar day) |
| **TGT\_QTY** | Actual quantity values of target variable |
| **STOCK\_QTY** | Stock qty (BOP) |
| **PROMO\_ID** | Promo id from sales/sellin/sellout table |
| **PROMO\_FLG** | Promo\_flg from sales/sellin/sellout table |

## Adding promo flag

**Inputs:** IN\_PROMO, T2

**Transformation algorithm:**

1. If for considered **tgt\_type** the value of the field **link\_with\_promo** is equal 1, then do steps below. Else the fields PROMO\_ID and PROMO\_TYPE should equal missings, PROMO\_FLG = 0.
2. T2 Left join IN\_PROMO on product, location, customer\_id, distr\_channel\_id, promo\_id and

IN\_PROMO.period\_start\_dt<= T2.period\_dt and

IN\_PROMO.period\_end\_dt >= T2.period\_dt,

Where T1.PROMO\_ID is not missing

Then group by quintuple PRODUCT\_ID, LOCATION\_ID, CUSTOMER\_ID, DISTR\_CHANNEL\_ID, PERIOD\_DT and recalculate fields

AVERAGE(TGT\_QTY) as TGT\_QTY

AVERAGE(STOCK\_QTY) as STOCK\_QTY

MAX(PROMO\_FLG) as PROMO\_FLG

MAX(PROMO\_ID) as PROMO\_ID

/\*add promo\_type that corresponds the promo with max id\*/

MAX(CASE WHEN PROMO\_ID = MAX(PROMO\_ID) THEN PROMO\_TYPE ELSE missing END) as PROMO\_TYPE

Assumption: there is only one promo\_id within each quintuple product\_id/location\_id/customer\_id/disr\_channel\_id/period\_dt.

and add promo\_type column to output table.

1. T2 left join IN\_PROMO on LOCATION\_ID, PRODUCT\_ID, CUSTOMER\_ID, DISTR\_CHANNEL\_ID and

IN\_PROMO.period\_start\_dt<= T2.period\_dt and

IN\_PROMO.period\_end\_dt >= T2.period\_dt,

Where T2.PROMO\_ID is missing

and updatePROMO\_ID and PROMO\_TYPE column with info from IN\_PROMO

PROMO\_TYPE = IN\_PROMO.PROMO\_TYPE

PROMO\_ID = IN\_PROMO.PROMO\_ID:

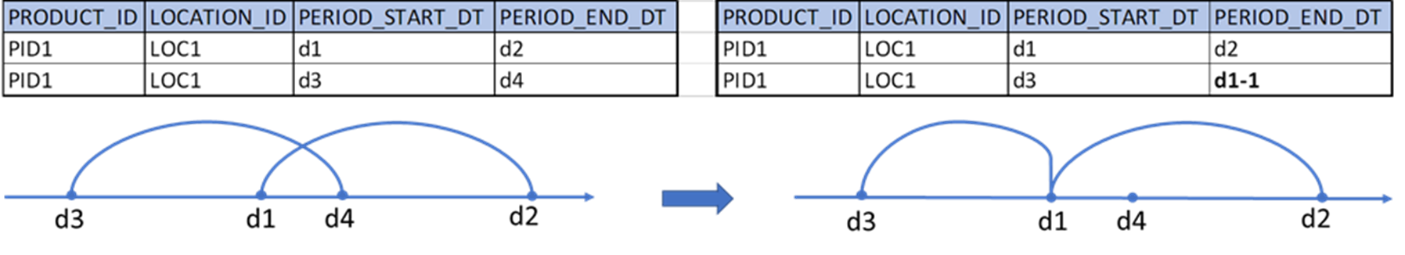
* 1. if there are several PROMO\_TYPEs for a triple product/location/customer/distr\_channel/day then select PROMO\_TYPE that related to the latest PERIOD\_START DT (see Figure 1 belowОшибка: источник перекрёстной ссылки не найден);

Figure : Interception of price periods

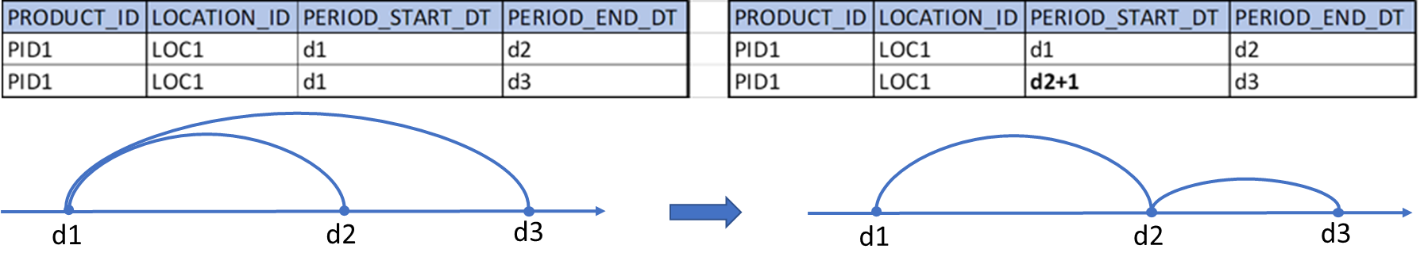
* 1. if there are several PROMO\_TYPEs for a quintupleproduct/location/customer/distr\_channel/day with the same PERIOD\_START\_DT then select PROMO\_TYPE that relates to the earliest PERIOD\_END\_DT (see Figure 2: Price Interception case 2 below);
     1. if there are still several promo types, choose maximal promo\_type.

Figure : Price Interception case 2

* + 1. Update PROMO\_ID Column with correspondent value

1. Union results from 2 and 3.

ASSUMPTION: if there is no related promo\_id or promo event for a quintuple then PROMO\_TYPE will be equal missing and PROMO\_ID will be missing.

ASSUMPTION: there can be cases when T2.promo\_flg = 0 but promo id and promo\_type are not missing

1. Define columns for the next steps

* TGT\_QTY\_R = TGT\_QTY
* DEFICIT\_FLG1 = 0
* DEFICIT\_FLG2 = 0
* COUNT\_NONDEFICIT\_DAYS = missing
* MEAN = missing
* STD = missing

Output: As a result of this step, a table of the following structure is constructed, *T3*

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| **PRODUCT\_ID** | Product ID (the lowest level of the product hierarchy) |
| **LOCATION\_ID** | Location ID |
| **CUSTOMER\_ID** | Customer ID |
| **DISTR\_CHANNEL\_ID** | Distribution Channel ID |
| **PERIOD\_DT** | Date of sales (calendar day) |
| **TGT\_QTY** | Actual quantity values of target variable |
| **STOCK\_QTY** | Stock qty (BOP) |
| **PROMO\_FLG** | 1| promo event was active  0| no promo event |
| **PROMO\_ID** | Promo ID |
| **PROMO\_TYPE** | Promo\_type id or missing if there is no promo event |
| **DEFICIT\_FLG1** | 1|primary deficit occurred  0|no primary deficit |
| **DEFICIT\_FLG2** | 1|secondary deficit occurred  0|no secondary deficit |
| **COUNT\_NONDEFICIT\_DAYS** | The number of non-deficit days (primary deficit) that were included in the window of **DR\_PERIOD\_LENGTH** previous calendar days (for particular group of days) |
| **MEAN** | Average number of sales values for the previous **DR\_OBS\_NUM** observations without primary deficit () within particular group of days |
| **STD** | Standard deviation of sales values for the previous **DR\_OBS\_NUM** observations without primary deficit ( within particular group of days |
| **TGT\_QTY\_R** | Restored demand for chosen target variable |

## Demand restoration algorithm

**Inputs:** T3

**Transformation algorithm:**

Demand restoration for each target variable is restored based on its scenario, which is defined using the field **dr\_scen** in the config file “tgt\_var\_config.csv”.

There are 4 scenarios, which differ in the set of steps for execution:

* Scenario #0. No demand restoration for chosen variable. Algorithm isn’t executed. Execution condition: parameter **dr\_scen** = 0 **OR** source table for TGT\_QTY is empty **OR** condition for other scenarios aren’t satisfied.
* Scenario #1. Demand restoration is being executed based on stocks, deficit periods and so on, but without history extending for short seasonal time-series. Only steps 4.4.1 - 4.4.3 are being executed. Execution condition: parameter **dr\_scen** = 1 **AND** source table for TGT\_QTY is not empty **AND** IN\_STOCK table is not empty **AND** parameter **link\_with\_stock** <> 0, else Scenario #0 is executed

Assumption: Customer has to check whether stock data is provided to the target var.

* Scenario #2. Only history extending for short seasonal time-series is performed. Only steps 4.4.4 are being executed. Execution condition: parameter **dr\_scen** = 2 and source table for TGT\_QTY is not empty **AND** seasonal indicator exists in product attributes data **AND** it was configured in related csv-file, else Scenario #0 is executed.
* Scenario #3. Cascade of Scenario #1 and Scenario #2. All steps 4.3.1 - 4.3.4 are being executed. Execution condition: parameter **dr\_scen** = 3 **AND** source table for TGT\_QTY is not empty **AND** IN\_STOCK table is not empty **AND** parameter **link\_with\_stock** <> 0 **AND** source table for TGT\_QTY is not empty **AND** seasonal indicator exists in product attributes data **AND** it was configured in related csv-file, else Scenario #0 is executed.

## Primary deficit flag definition

This step is executed if dr\_scen = 1 or 3

**Inputs:** T3

**Transformation algorithm:**

1. Identify periods when a store was temporary closed based on T3:
   1. Calculate SUM\_TGT\_QTY as sum (TGT\_QTY) by LOCATION\_ID | PERIOD\_DT.
   2. Mark days when store was closed as
   3. T3 left join the table from step a. on LOCATION\_ID | PERIOD\_DT and add column CLOSED\_FLG to T3.
2. The primary deficit flag (DEFICIT\_FLG1) is calculated for each combination of product | location | customer | distribution channel | day using the following rule:

**Output:** as a result of this step, a table of the following structure is constructed, *T41*

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| **PRODUCT\_ID** | Product ID (the lowest level of the product hierarchy) |
| **LOCATION\_ID** | Location ID |
| **CUSTOMER\_ID** | Customer ID |
| **DISTR\_CHANNEL\_ID** | Distribution Channel ID |
| **PERIOD\_DT** | Date of sales (calendar day) |
| **TGT\_QTY\_R** | Actual quantity values of target variable |
| **STOCK\_QTY** | Stock qty (BOP) |
| **PROMO\_FLG** | 1| promo event was active  0| no promo event |
| **DEFICIT\_FLG1** | 1| primary deficit occurred  0| no primary deficit |
| **PROMO\_ID** | Promo ID |
| **PROMO\_TYPE** | Promo\_type id or missing if there is no promo event |
| **TGT\_QTY\_R** | Restored demand for chosen target variable |

## Secondary deficit flag definition and mean calculation

This step is executed if dr\_scen = 1 or 3

**Inputs:** T41

**Transformation algorithm:**

1. For each product | location | customer | distribution channel combination, divide T3 into groups: 1) regular demand days (where promo\_type is equal to missing); 2) promo type 1 days (where type is equal to 1), 3) promo type 2 days etc (depending on how many promo types there are).

For each product | location | customer | distribution channel and each group of days on each date since (>=) If **IB\_UPDATE\_HISTORY\_DEPTH<=0 then IB\_HIST\_START\_DT else IB\_HIST\_END\_DT - IB\_UPDATE\_HISTORY\_DEPTH**:

* Select only those days which belongs to

period.

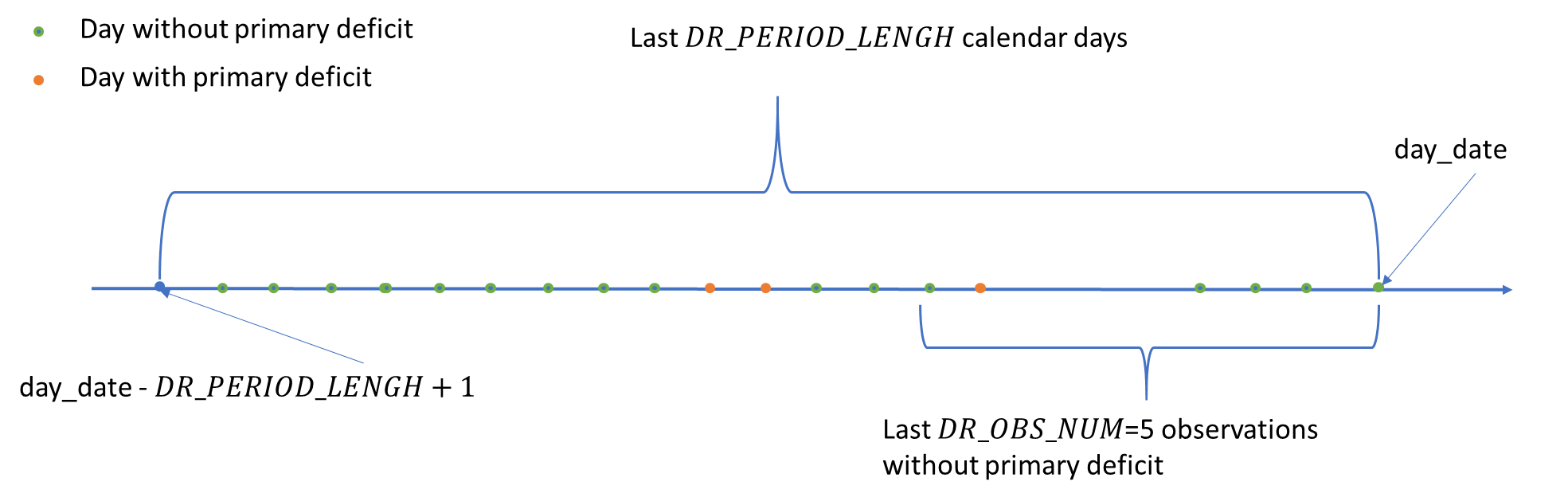
* Then calculate the average value[[3]](#footnote-3) (and the standard deviation value[[4]](#footnote-4)) and count of observations (**COUNT\_NONDEFICIT\_DAYS**) of sales based on no more than last DR\_OBS\_NUM non-deficit observations (without a primary deficit) from day\_date (see picture Figure 3 below).

Figure : Two-step filtering approach for calculating mean and std

1. Fill missing value: For each product | location | customer | distribution channel and each group of days on each date since (>=) If **IB\_UPDATE\_HISTORY\_DEPTH<=0 then IB\_HIST\_START\_DT else IB\_HIST\_END\_DT - IB\_UPDATE\_HISTORY\_DEPTH**
   1. If the values of the mean (or standard deviation) is missing for a day, then fill them in with the previous[[5]](#footnote-5) non-missing mean (standard deviation) value[[6]](#footnote-6), if there is no previous non-missing value fill mean value with 0 (standard deviation with 1).
2. For each product | location | customer | distribution channel and each group of days on each date, the following should be calculated:

* The threshold is defined by the formula:

1. For each quadruple product | location | customer | distribution channel and each group of days on each date since (>=) If **IB\_UPDATE\_HISTORY\_DEPTH<=0 then IB\_HIST\_START\_DT else IB\_HIST\_END\_DT - IB\_UPDATE\_HISTORY\_DEPTH** define the secondary deficit flag according to the following rules:

* If there was a primary deficit on that day, there was a secondary deficit as well;
* If TGT\_QTY at this day is less than (see above), then the secondary deficit took place;
* If **HIGH\_TURNOVER\_TRSHD** (high-turnover quadruple) and volume of sales on this day TGT\_QTY is less then , then the secondary deficit took place;
* Otherwise, there was no secondary deficit;

1. Union results for all group of days

**Output:** as a result of this step, a table of the following structure is constructed, *T42*

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| **<T41>** |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Demand restoration based on stock deficit

This step is executed if dr\_scen = 1 or 3

**Inputs:** T42

1. At this stage, the historical demand is restored according to the following rule for each date since (>=) If **IB\_UPDATE\_HISTORY\_DEPTH<=0 then IB\_HIST\_START\_DT else IB\_HIST\_END\_DT - IB\_UPDATE\_HISTORY\_DEPTH**:

**Output:** as a result of this step, a table T43 of the following structure as for T42 but with additional field TGT\_QTY\_R is constructed.

## Update RESTORED DEMAND table

**Inputs:** T43 or T3 (depending on dr\_scen value)

1. Update Restored Demand Table with new values
   1. SELECT \*

FROM RESTORED DEMAND

WHERE period < **IB\_HIST\_END\_DT - IB\_UPDATE\_HISTORY\_DEPTH**

**UNION**

SELECT \*

FROM 1

WHERE period >= IFN (**IB\_UPDATE\_HISTORY\_DEPTH>0; IB\_HIST\_END\_DT - IB\_UPDATE\_HISTORY\_DEPTH, IB\_HIST\_START\_DT)**

**Output:** as a result of this step, a table DEMAND\_RESTORED of the following structure as for T42 but with additional field TGT\_QTY\_R is constructed (see section 3.5.).

## Demand history extending for short and seasonal time series

**Inputs:** RESTORED\_DEMAND, FORECAST\_FLAG, DDS.IN\_PRODUCT\_ATTR

At this stage, the historical demand is being extended for seasonal time series with insufficient history.

**Transformation algorithm:**

1. Import of config-file seasonal\_flag\_config.csv and do:
   1. Select name of the seasonal indicator from fromSeasonalFlag column by filter toSeasonalFlag = ‘SEASONAL\_FLAG’.
   2. Select list of values from fromSeasonalFlag column which means seasonal relation by filter toSeasonalFlag = ‘1’.
   3. Select list of products from DDS.IN\_PRODUCT\_ATTR where PRODUCT\_ATTR\_NAME equals seasonal indicator from step **a.** and PRODUCT\_ATTR\_VALUE equals values from step **b**.
2. For each time-series product | location | customer | distribution channel combination, where product is from list above (step 1), the depth of the demand history in months should be calculated as number of months between minimal (MIN\_MONTH\_DT) and maximal (MAX\_MONTH\_DT) from FORECAST\_FLAG table

SELECT PRODUCT\_ID, LOCATION\_ID, CUSTOMER\_ID, DISTR\_CHANNEL\_ID

, MIN(intnx(‘month’, period\_start\_dt, 0)) as MIN\_MONTH\_DT

, MAX(intnx(‘month’, MIN(period\_end\_dt, IB\_HIST\_END\_DT), 0)) as MAX\_MONTH\_DT

, date\_diff(‘month’, MIN\_MONTH\_DT, MAX\_MONTH\_DT) as HISTORY\_DEPTH

1. DEMAND\_RESTORED filter only quadruples from 1 and left join 2 on \*\*\_id.
2. For those quadruples from 3 which HISTORY\_DEPTH belongs to interval between **MIN\_PROLONG\_HIST\_MONTH** and **MAX\_PROLONG\_HIST\_MONTH**, and depth between MAX\_MONTH\_DT and today() less than 12 months then prolongation logic is being executed:
   1. Calculate median (per day) values of TGT\_QTY\_R through available history for each product | location | customer | distribution channel combination.
   2. Create 2 years calendar: from today() to today() – 2 years.
   3. Fulfill demand TGT\_QTY\_R during interval [MIN\_MONTH\_DT – 12 months, MAX\_MONTH\_DT – 12 months] with corresponding demand values TGT\_QTY\_R from interval [MIN\_MONTH\_DT, MAX\_MONTH\_DT].
   4. Fulfill remained dates from 2 years calendar as median values (step a.)

Else :

* for products NOT from the list in step 1,
* or for products from the list in step 1, but with sales history less than **MIN\_PROLONG\_HIST\_MONTH** or more than **MAX\_PROLONG\_HIST\_MONTH** or depth between MAX\_MONTH\_DT and today() greater or equal than 12 months**,**

this algorithm should not be performed.

1. Update DEMAND\_RESTORED:

* SELECT \* FROM DEMAND\_RESTORED WHERE \*\*\_ID not in list 1

UNION SELECT \* FROM **3**

Note: steps 4.4.5.3 – 5 lead to continuous updating DEMAND\_RESTORED table regarding dr\_scen #3.

**Output:** as a result of this step, a table of the following structure is constructed, *DEMAND\_RESTORED see description at section 4.5.*

## Output from the algorithm

Name of the output tables of demand restoration algorithm should include postfix with type of target variable based on the field tgt\_type in configuration-file “tgt\_var\_config.csv”: \_SELLIN, \_SELLOUT or \_POS.

Final output table should look like as follows:

|  |  |
| --- | --- |
| DEMAND\_RESTORED \_<postfix for considered target variable type> | |
| Field | Description |
| **PRODUCT\_ID** | Product ID |
| **LOCATION\_ID** | Location ID |
| **CUSTOMER\_ID** | Customer ID |
| **DISTR\_CHANNEL\_ID** | Distribution Channel ID |
| **PERIOD\_DT** | Date of sales (calendar day) |
| **TGT\_QTY** | Actual quantity values of target variable |
| **TGT\_QTY\_R** | Restored demand for chosen target variable |
| **STOCK\_QTY** | Stock qty (BOP) |
| **PROMO\_FLG** | 1|promo event was active  0|no promo event |
| **PROMO\_TYPE** | Promo\_type id or missing if there is no promo event |
| **PROMO\_ID** | Promo ID |
| **DEFICIT\_FLG1** | 1|primary deficit occurred  0|no primary deficit |
| **DEFICIT\_FLG2** | 1|secondary deficit occurred  0|no secondary deficit |
| **COUNT\_NONDEFICIT\_DAYS** | The number of non-deficit days (primary deficit) that were included in the window of **DR\_PERIOD\_LENGTH** previous calendar days |
| **MEAN** | Average number of sales values for the previous **DR\_OBS\_NUM** observations without primary deficit () within particular group of days |
| **STD** | Standard deviation of sales values for the previous **DR\_OBS\_NUM** observations without primary deficit ( within particular group of days |

1. It doesn’t mean that global parameter must by rewritten, it means that parameter value must be different for particular run. [↑](#footnote-ref-1)
2. It is assumed final true information regarding product lifecycle to be stored in FORECAST\_FLAG table. [↑](#footnote-ref-2)
3. Considering that average and standard deviation is created for each date we may leverage moving average and moving standard deviation algorithms. [↑](#footnote-ref-3)
4. [↑](#footnote-ref-4)
5. Search for previous non-missing value within the whole past period (i.e. there is no time range limitation while searching previous non-missing value). [↑](#footnote-ref-5)
6. It may be a case that for some days the is no previous correspondent mean (standard deviation) value, then final [↑](#footnote-ref-6)